FINAL PROJECT

Rayan Jaipuriyar

Northwestern University

MSDS 458: Artificial Intelligence and Deep Learning

Professor Syamala Srinivasan

August 23, 2024

Table of Contents

Abstract

This paper explores variations in the architecture of a Deep Learning Neural Network designed to recognise and classify images of Stars and Galaxies. The input data took the form of an images with 64*64*3 pixels. It was determined that a model with three convolution layers, and one dense layer, performs at the optimal accuracy when it has Ridge regularization and drop out layers implemented within it.

Introduction

We are an independent contractor hired by the Northwestern Astronomy Department to train a model that can recognise astronomical images. Telescopes often collect a vast array of data with minimal oversight through the use of automated programs that scan the sky. However, combing through the collected images requires a lot of time and manpower. This presents a significant challenge for University researchers, as a lack of human resources adversely affects the extraction of accurate information from these vast data sets. Therefore, to address this issue, we aim to develop a robust and accurate model that is capable of reliably interpreting collected images from telescopes with a high prediction accuracy. To justify the use of a Deep Neural Network over a human interpreter, we will require the model's accuracy to be above 70%.

Literature Review

Image recognition has been a subject of much interest over several years. With advancements in Neural Networks, models classifying images have improved significantly. CNNs have been used to recognize handwriting and have been shown to have predictive accuracies of as high as 99.22% (Ali et al., 2019). Deep neural networks have also been used to interpret the emotions of a subject in an image (Kim et al., 2018). We aim to replicate these techniques to classify celestial objects in the far Universe.

Methodology

We will be building a series of Deep Neural Networks that will be trained on a dataset that contains a mix of images containing Galaxies and Stars. The original data was sourced from Kaggle and contained 942 Galaxy images and 3044 Star images. For the sake of training an impartial model, the Star images were reduced to 942 files via the use of a random state. The images in these data sets were sorted into two different files. Each image had a size of 64x64 pixels. The data set was split using a random state to have 377 images with an equal number of Galaxies and Stars as the testing data set, leaving 1507 images in the training data set. The training data was further split to create a Validation data set, with 150 images that will be used to

track accuracies across during training. The pixels of the image arrays can take a value between 0 and 255, depending on the intensity of the color black for the given pixel. We will normalize the image data so that they take values between 0 and 1 instead of 0 and 255. An example image can be viewed in Appendix A.

We will explore a range of different network architectures in order to optimize the model, while limiting overfitting by having too many filters and nodes. We will rely heavily on convolution layers to interpret the images, with each convolution layer followed by a pooling layer that condenses the filtered data in a square. This process is repeated for every filter in the convolution layer. The convolution layer is then followed by a dense layer that will attempt to recognize the image. An example model architecture can be found in Appendix B. We will construct the CNN using Keras and TensorFlow. The TensorFlow version we are using is 2.16.1. TensorFlow inbuilt functionality will be used to perform feature extraction. In addition, we will make use of Scikit-learn to construct confusion matrices. Matplotlib and Seaborn are also called to create visualizations. Finally, Numpy and Pandas are used for handling the arrays and data frames.

Results

Experiments 0, 1, and 2

As a first step, we will test a simple dense network with varying architectures on our data set to ascertain if this model type can make for a suitable classifier for the said data. We ran three experiments. The first with a single, 128 node dense layer. The second with two dense hidden layers with 128 nodes and 100 nodes respectively. And the third with three hidden layers, each with 128, 100, and 100 hidden layers respectively. The accuracy of the models when run on the test data set are as follows:

Experiment 0: 63.50%
Experiment 1: 53.82%
Experiment 2: 54.38%

It is clear from these results that a dense network is not a suitable network architecture for the classification problem we are trying to address. Infact, increasing model depth hampers the models accuracy. Additionally, it was ascertained through testing that increasing the node numbers of these models does not improve accuracy in any way. Therefore we will swiftly move forward with testing CNNs in the following experiments, as they are more suited for image classification tasks.

Experiment 3

In this experiment, we will build a model with two convolution hidden layers prior to a dense hidden layer. All the convolution layers will have a Relu activation. The first layer will have 128 filters, with a kernel size of 5x5, and a stride size of 1 pixel. Initially a kernel size of 3x3 was chosen however, this frame proved to be too small for the model to extract the necessary features, resulting in a feedback loop that kept model validation and training accuracy low. The second layer will have 256 filters with the same kernel and step sizes as that of the previous layer. Each convolution layer will be followed by a max pooling layer of pool size 2x2. The convolution layers will be followed by a single dense layer with 128 nodes and a SoftMax activation function, followed by 2 output nodes. The performance metrics of this model is as follows:

Training data accuracy: 93.94%
Validation data accuracy: 88.00%
Testing data accuracy: 84.51%

We can see that this model is a significant improvement from the previous DNN model. Seeing as this is the first model that gives us passable results, further study of the models metrics and features is warranted. A graph of training and validation data accuracies across epochs can be found in Appendix F. It is evident that overfitting is still an issue that persists in this model. At some point we may need to introduce a regularization technique to address this issue. The confusion matrix (which can also be found in Appendix F) shows that this model has a similar predilection to over-classifying images as stars with 40 galaxies erroneously classified as stars and 23 stars erroneously classified as galaxies. A catalog of the first 20 galaxies classified as stars, and the first 20 stars classified as galaxies can be found in Appendix F.

From a visual inspection it appears that objects that appear more circular in the image are predisposed to be classified as stars by the model, while elliptical, or oblong objects are more likely to be classified as galaxies. We can test to see if this hypothesis is correct by conducting feature extraction on one of the images in the wrongly classified Galaxies group. The extracted features of one of the images, collected from the first 4 layers of our model are stored in Appendix F. While some of the initial filters do extract oblong shapes from the image in question, by layer 4 it is clear that most of the filters have saved a more rotund shape, which perhaps leads to the misclassification. On the other hand, misclassified stars appear to be classified due to elliptical and oblong features being extracted by the model (Appendix F, next page).

Additionally, binary systems (systems with two objects in the same image) are well represented in the wrongly classified image group implying that the model does not do a great job of extracting features for these types of images. A comparison of the final max pooling

layers' feature extraction can be found in Appendix F. The binary nature is visible in both extracted features, making them pretty similar to the model, from a computer vision standpoint. This may be why the model does badly in classifying them since it must think they are the same type of object.

It would seem we are on the right track by using a CNN with the specified parameters, however, improvements can be made to the architecture to correct for some of the outlined shortcomings of this model. We will attempt this by adding another set of convolution and max pooling layers between layer 2 and the dense layer 3. Additionally, from the graph of training and validation data accuracies across epochs, it can be ascertained that our model is overfitting to the training data. We will need to correct for this in future iterations of the model.

Experiment 4

We will improve on the previous model by adding another set of convolution and max pooling layers between layer 2 and the dense layer 3. The new convolution layer will have 512 filters and a 5x5 kernel size. In addition we will include a batch normalization layer after the dense layer, to improve computational time. The accuracies of the model on the training, validation, and testing data is as follows:

Training data accuracy: 85.78%
Validation data accuracy: 88.00%
Testing data accuracy: 86.82%

While accuracy has not changed significantly for our new model, The confusion matrix in Appendix G shows us that fewer misclassifications are made by this model. It is also evident that our overfitting problem is not going away. This again, is made all the more clear when inspecting the graph comparing training and validation model accuracies across epochs (Appendix G). What is more worrying is that a concerning variability in accuracy across epochs can be seen in this new model. This phenomenon was not present in our previous model. To address this, we reran this experiment, varied the final convolution layers kernel size, in the hopes that a smaller kernel size would allow for the inclusion of smaller, more localized features, however, this did not result in substantial model improvement. Clearly the inclusion of another set of convolution/max pooling layers has not helped address the overfitting in our model. Therefore, we shall implement regularization techniques to this model in the next experiment.

Experiments 5

In this experiment we will attempt to combat the overfitting issue our model has been running into by introducing regularization and dropout layers. be adding a regularization technique to the three layer CNN model from the previous model experiment. We will be using

an L2 or Ridge regularization technique that adds a penalty proportional to the sum of the squares of the weights in each node in the dense layer of the network (Google, 2022). This encourages the model to keep the weights small and prevents over-specialization. In addition, we will add a drop out layer after every convolution/max pooling layer and dense layer. These drop out layers will turn off 30% of the prior layer's filters/nodes. This prevents any one node from becoming overly specialized, since when a specialized node is turned off the remaining nodes will need to fill in the gaps created. All other parameters, including the filter/node number, remain the same as they were in the previous experiment. The accuracy measures for this experiment is as follows:

Training data accuracy: 88.16%
Validation data accuracy: 90.67%
Testing data accuracy: 82.37%

While the validation accuracy has shown a significant improvement, the testing accuracy has dipped for this model. Taking a look at training, validation accuracy across epochs in Appendix H, it is evident that our strategies to counter overfitting have helped reduce the gap between the training and validation data sets. However the instability present in the validation accuracy during training is still present in this model. This instability is due to the fact that our training data is relatively small, resulting in adaptations that are not general enough to show consistent improvements.

Experiment 6

With this in mind, we can conclude that the model in Experiment 5 is the best of all the tested options so far, given the circumstances. We implemented changes to this model's architecture. Variations on the number of filters and nodes in each layer were tested and it was found that a model architecture with 128 filters in each of the three convolution layers and 300 nodes in the dense layer resulted in the least amount of overfitting. A summary of this model can be found in Appendix I The accuracy measures of this model are as follows:

Training data accuracy: 82.80%
Validation data accuracy: 86.67%
Testing data accuracy: 85.56%

Comparing all the model iterations tested so far, the model in this experiment gives us the least amount of overfitting while holding a competitive accuracy. A look at the misclassified images in Appendix I shows us that the model does not seem to show a significant bias towards classifying images as Stars. Additionally, a significant number of the images that have been misclassified are the binary star/galaxy images, with more than one celestial object in the image.

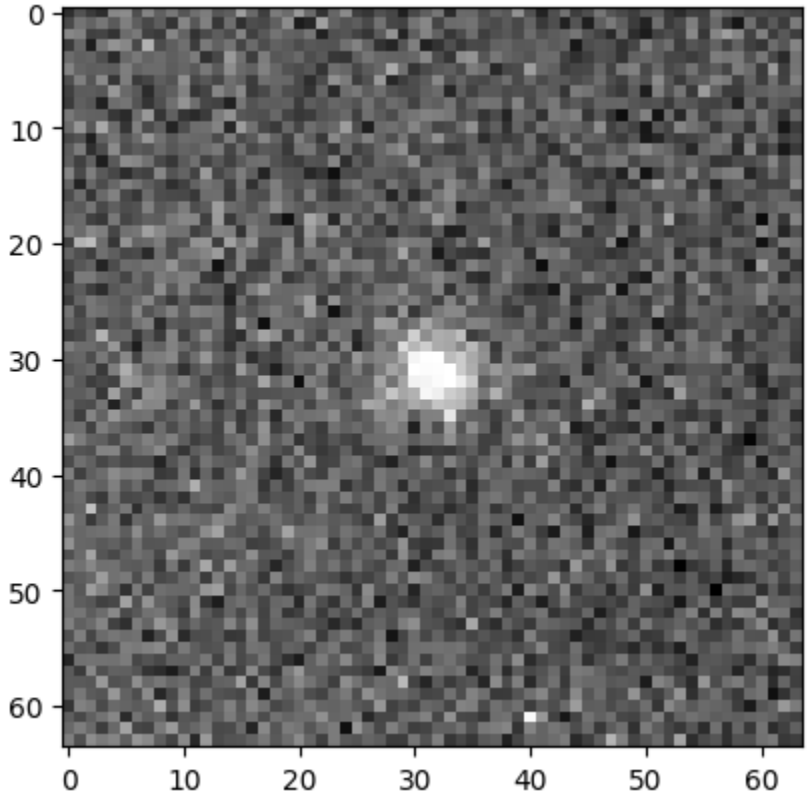
We have discussed why the model has trouble accurately classifying these types of images in Experiment 3, and so these misclassifications are to be expected.

Conclusion

We have constructed several models with varying architectures that consist of a combination of convolution and dense layers. These models were trained to recognise celestial entities from a collection of images of Stars and Galaxies. The accuracies of these models on training, validation and testing data are summarized in a table in Appendix J. A model with the following architecture was found to perform the best: Three convolution layers, with Relu activation, each followed by a max pooling layer. Each of the convolution layers has a kernel of size 3 by 3 and a step size of 1 pixel. The pooling layers have a pooling size of 2x2, and stride of 2. All convolution layers have 128 filters. After the convolution layers, there is a flattening layer that flattens the incoming arrays of data from a 64x64x3 array to a single dimensional array that can be fed into a DNN. The next layer is a dense layer with a SoftMax activation and a L2 regularizer with 300 nodes. We included a batch normalization layer after the input dense layer to reduce computational time, then followed up with the inclusion of a 2 node dense output layer, again with a SoftMax activation. Each layer of this model has 30% of its filters/nodes turned off every epoch via dropout layers after every convolution and dense layer. We were able to maximize the number of nodes in our dense layers without succumbing to overfitting by including L2/ridge regularization and the dropout layers. A summary of the described model can be found in Appendix I. We recommend the implementation of this model, since it outperforms all other tested models in this paper, while displaying minimal overfitting issues, and taking the same amount of computational time to train.

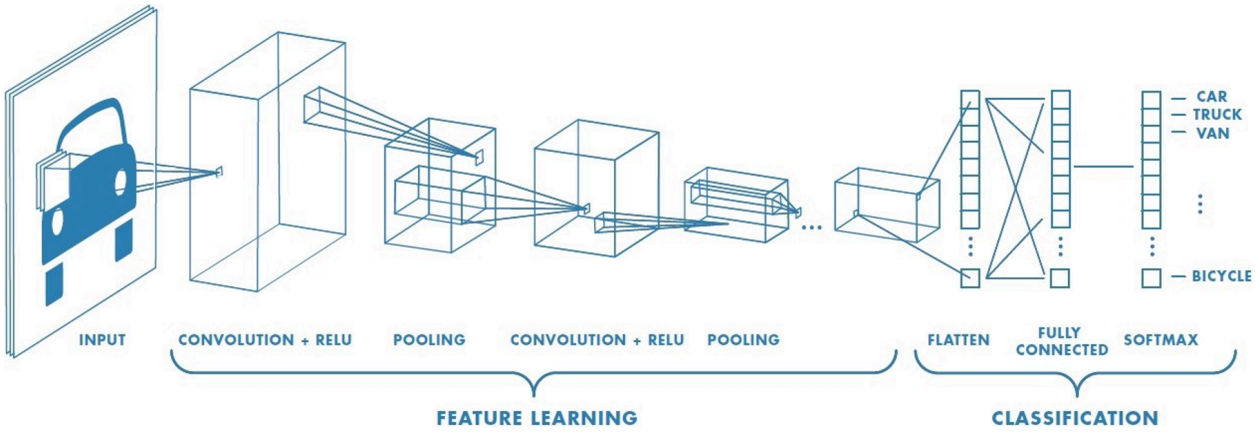
Appendix A

An example image in the data.



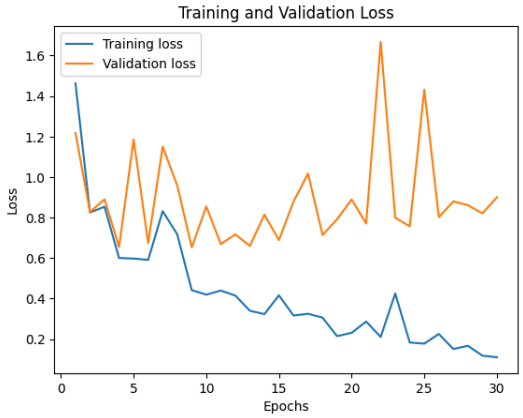
Appendix B

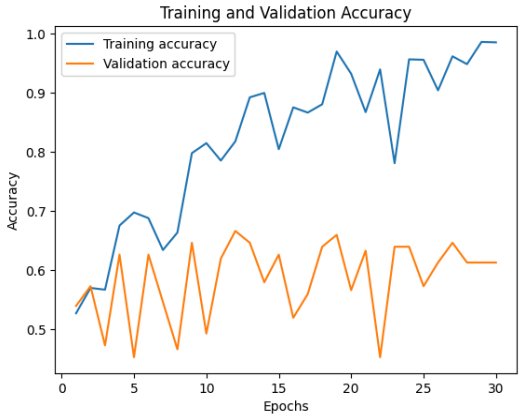
Example Structure of the CNN.



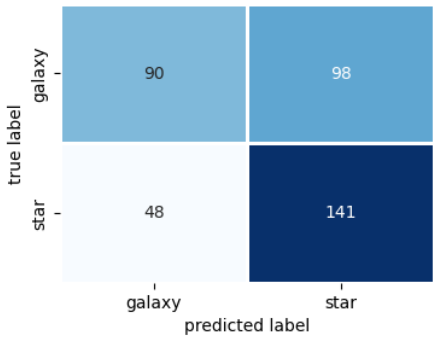
Appendix C

Graphs and visualizations for Experiment 0:



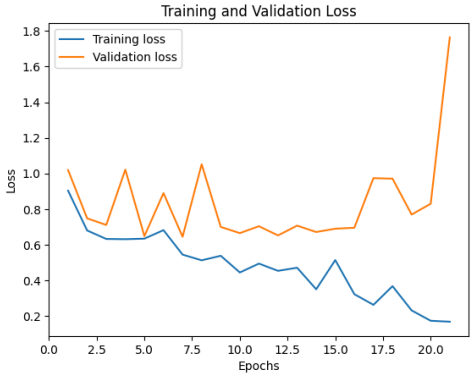


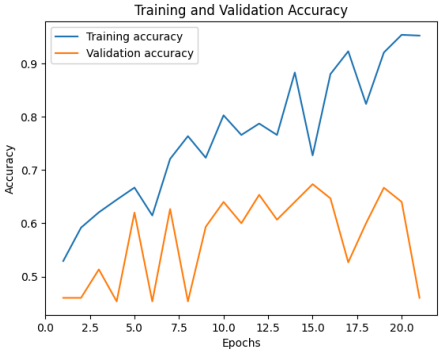
Confusion Matrix



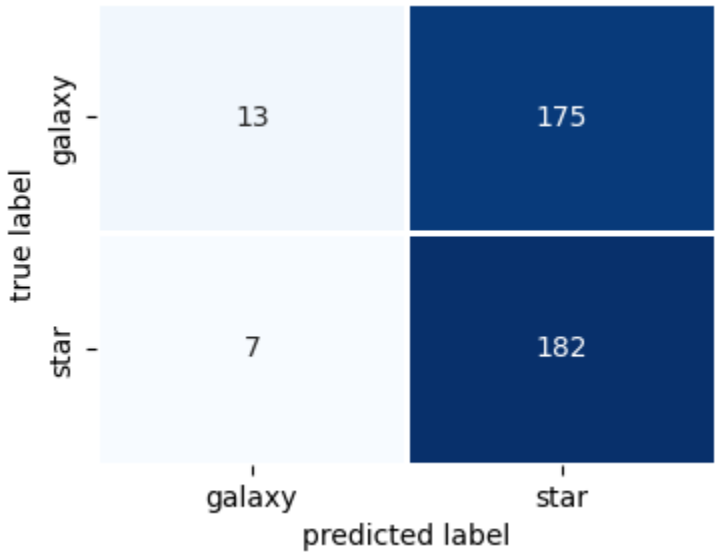
Appendix D

Graphs and visualizations for Experiment 1:



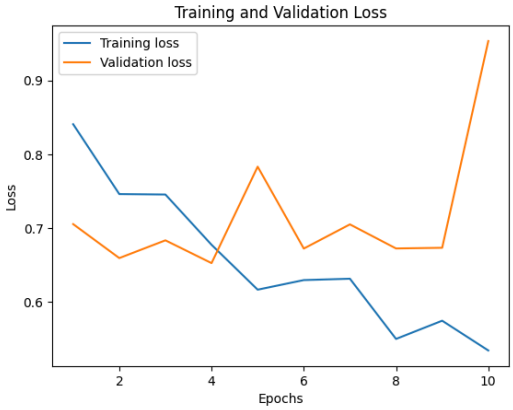


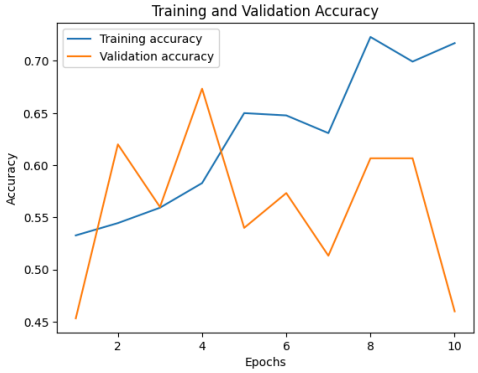
Confusion Matrix



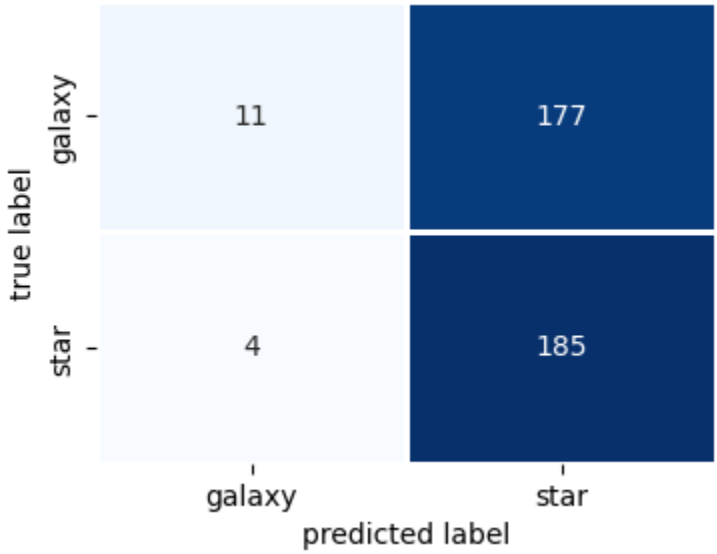
Appendix E

Graphs and visualizations for Experiment 2:



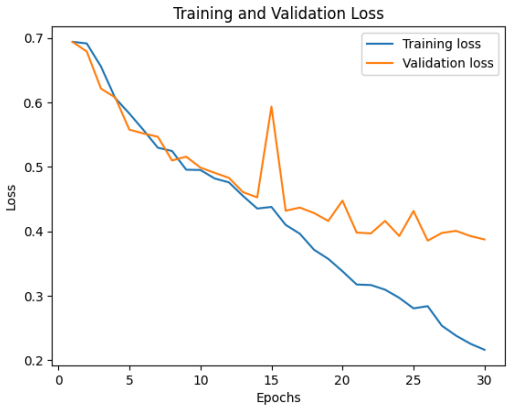


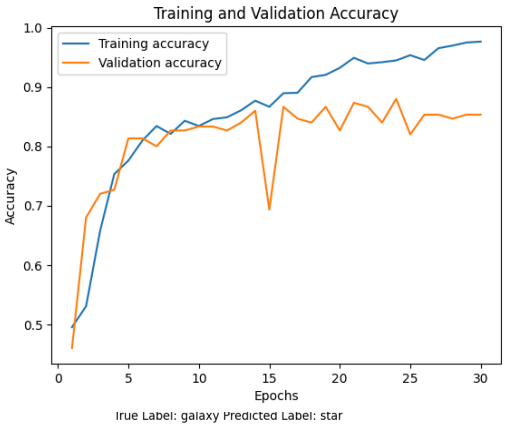
Confusion Matrix



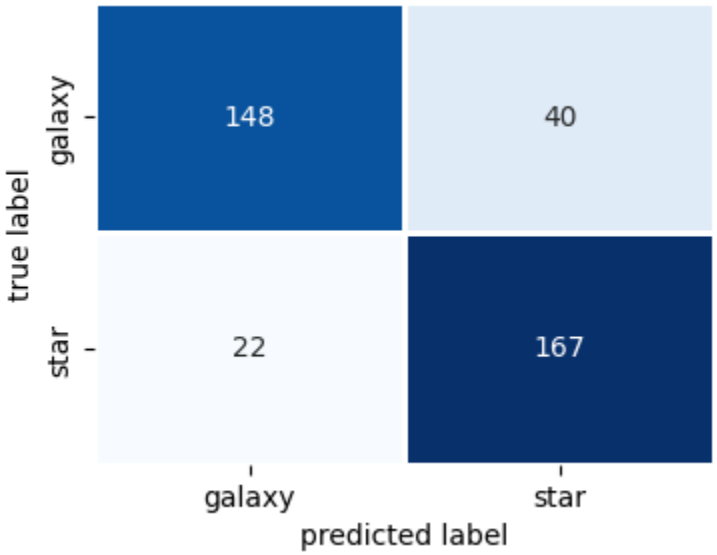
Appendix F

Graphs and visualizations for Experiment 3:



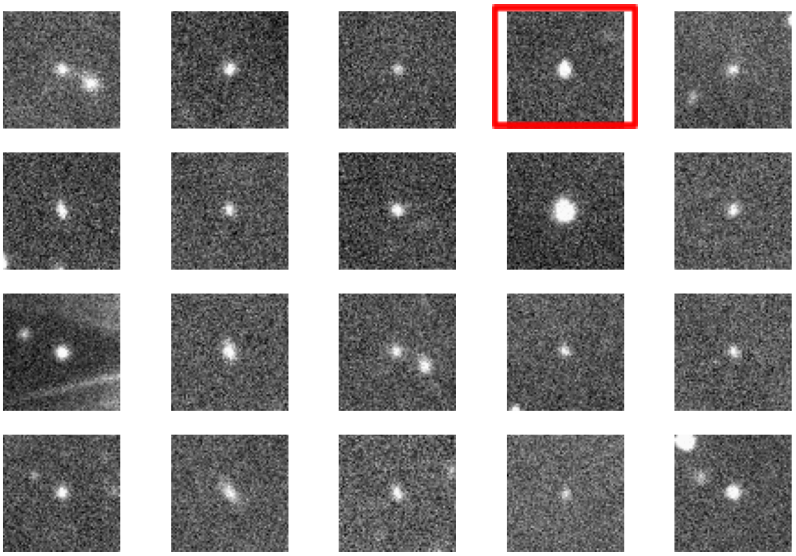


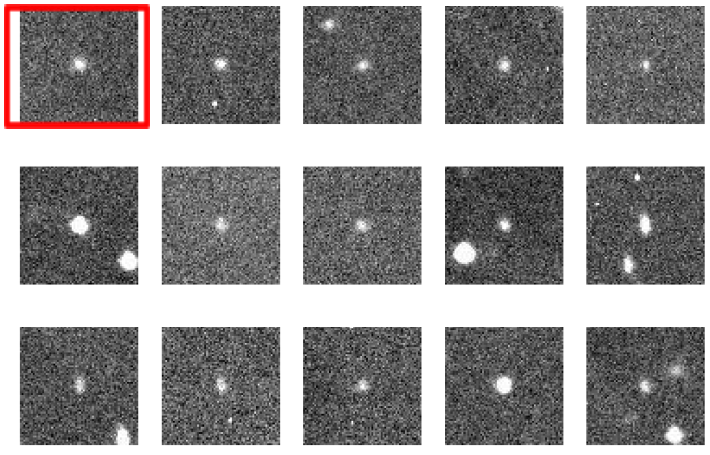
Confusion Matrix



True Label: galaxy Predicted Label: star

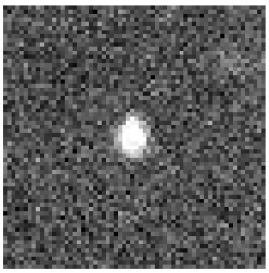
True Label: star Predicted Label: galaxy

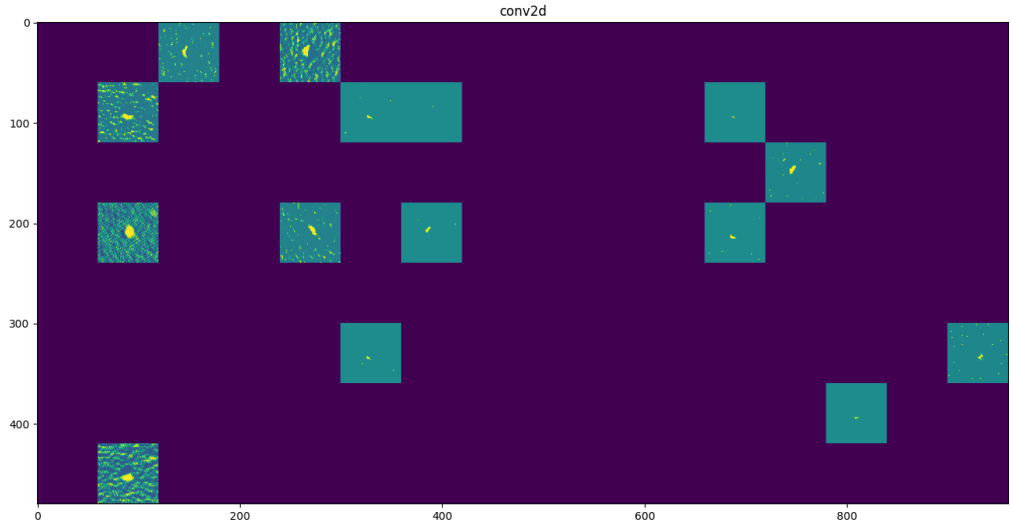


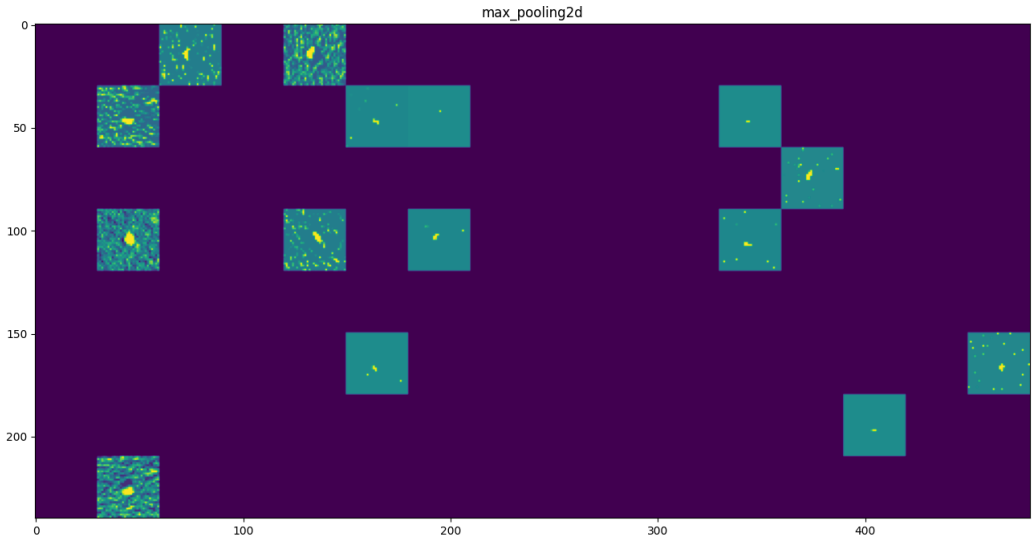


Feature Extraction for Galaxy Classified as Star:

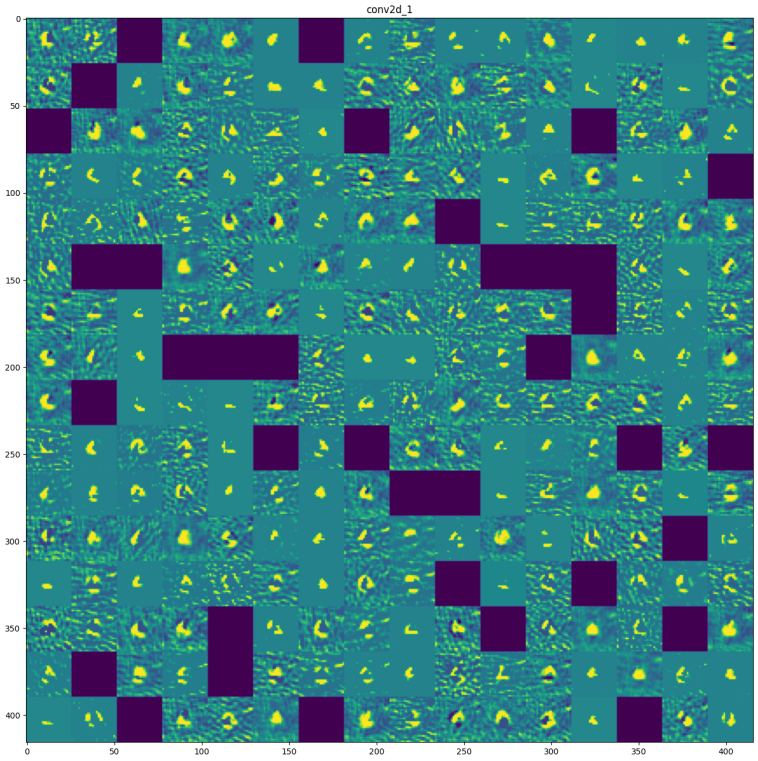
First Convolution and Max Pooling layers

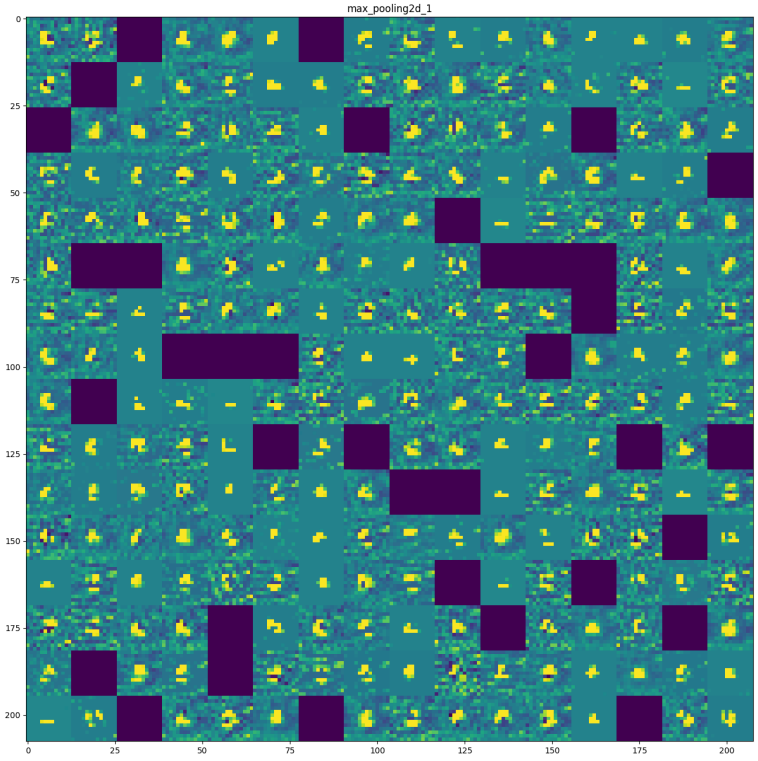






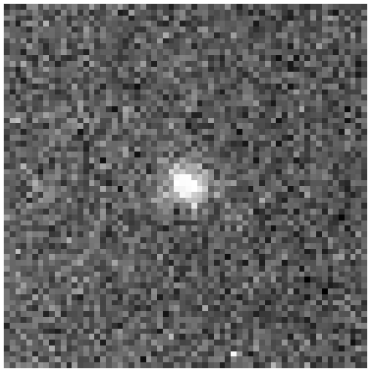
Second Convolution and Max Pooling layers

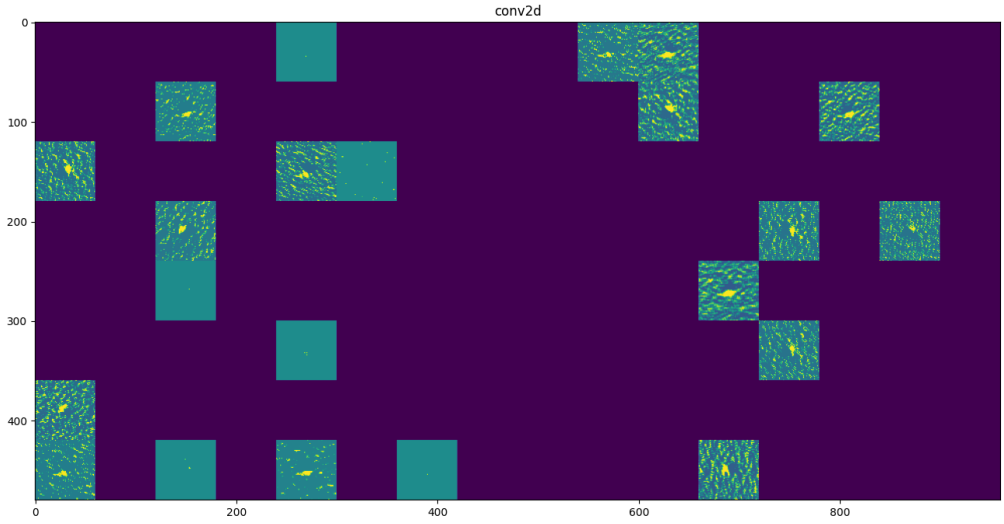


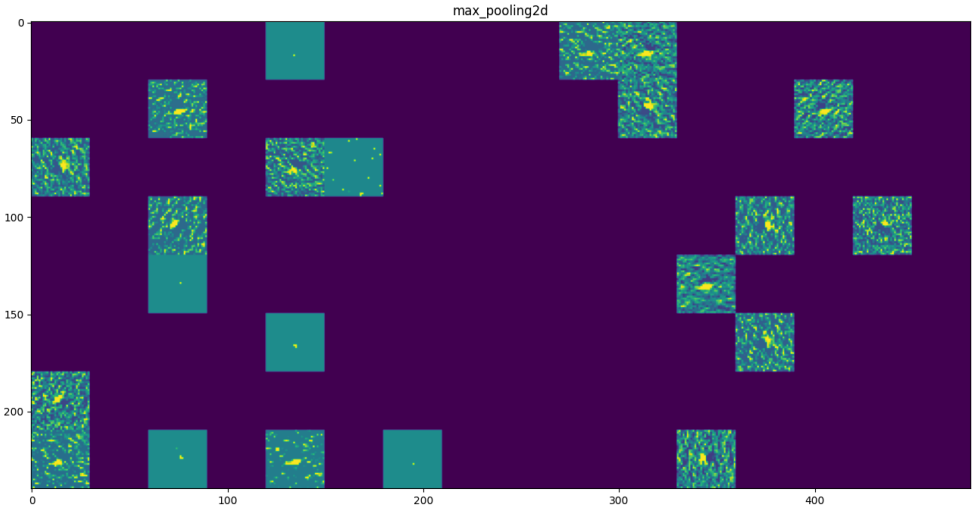


Feature Extraction for Star Classified as Galaxy:

First Convolution and Max Pooling layers

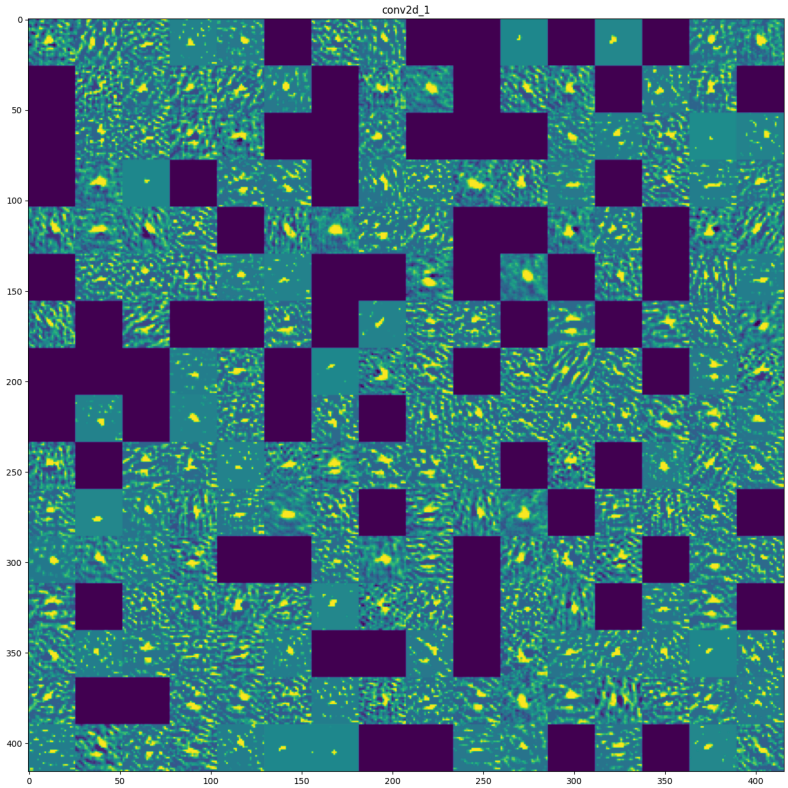


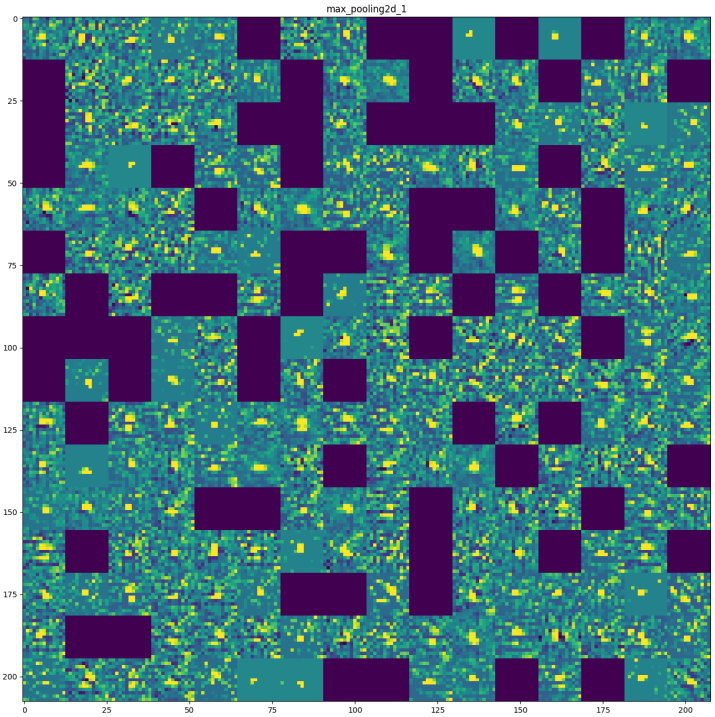




Feature Extraction for Star Classified as Galaxy:

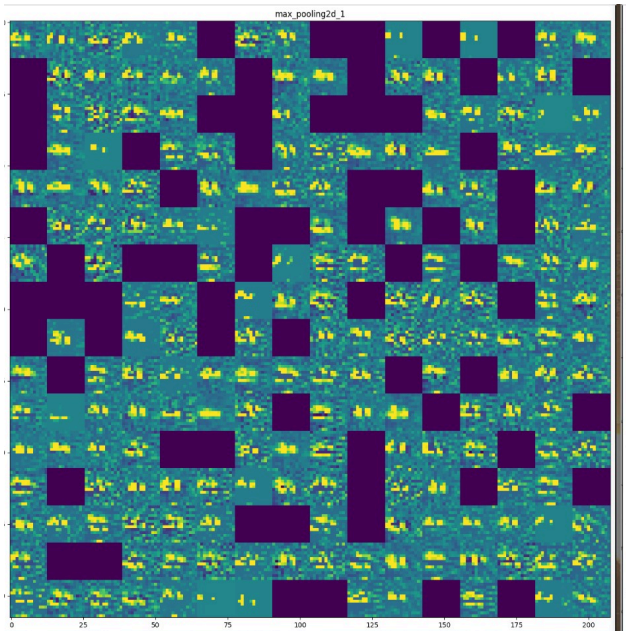
Second Convolution and Max Pooling layers



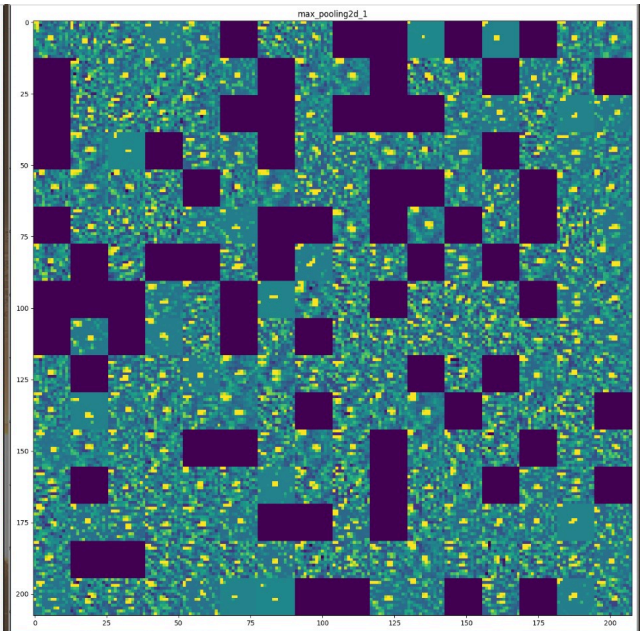


Final Pooling layer of Binary Misclassifications:

Misclassified Galaxy

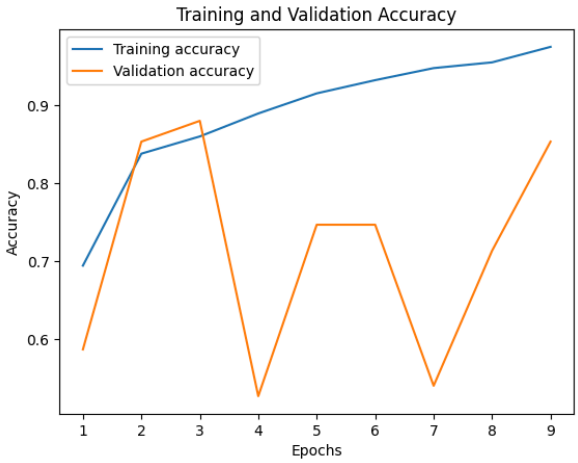


Misclassified Star

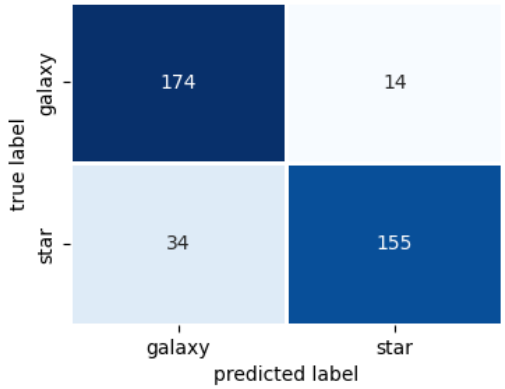


Appendix G

Graphs and visualizations for Experiment 4:

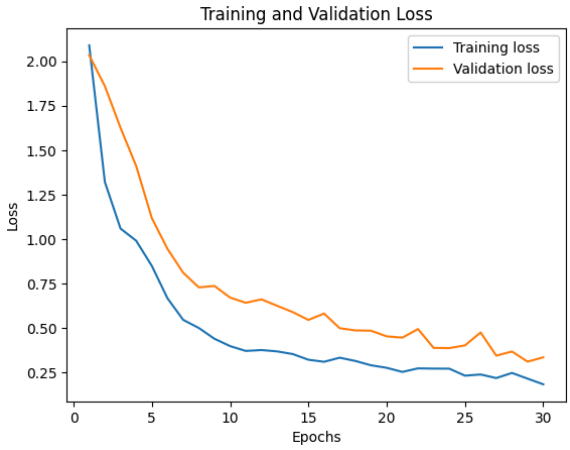


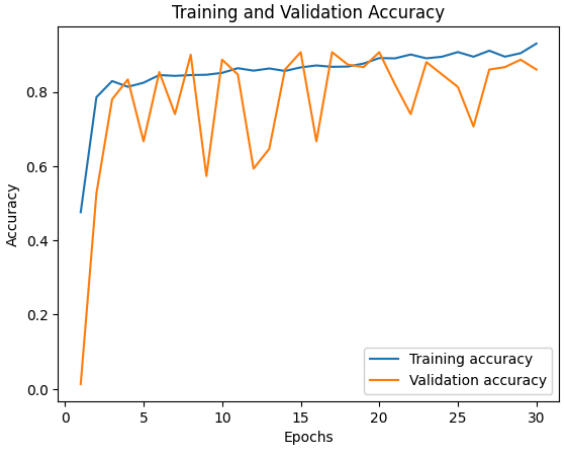
Confusion Matrix:



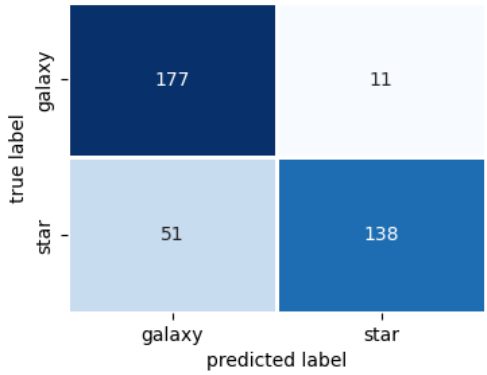
Appendix H

Experiments 5





Confusion Matrix:



Appendix I

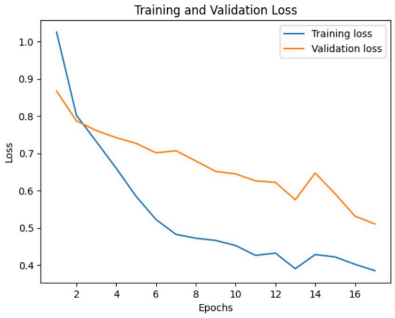
Summary, graphs, and visualizations for Experiment 6:

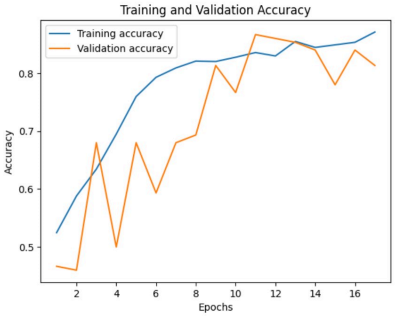
```
model.summary()
```

Model: "sequential"

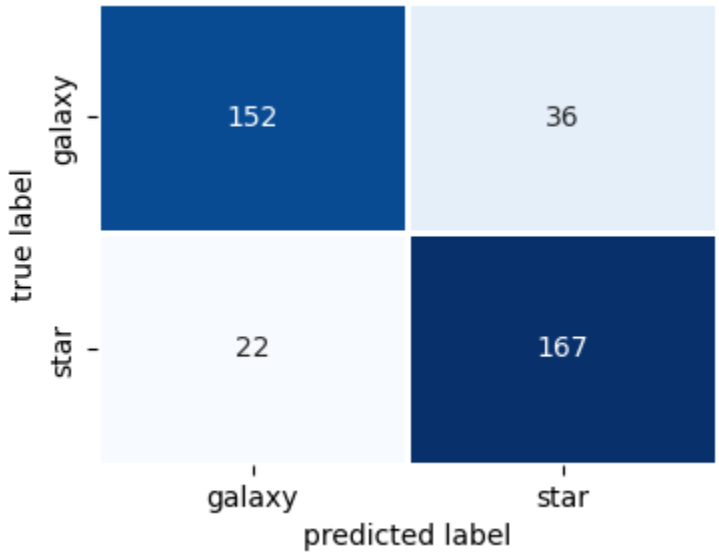
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 128)	3,584
max_pooling2d (MaxPooling2D)	(None, 31, 31, 128)	0
dropout (Dropout)	(None, 31, 31, 128)	0
conv2d_1 (Conv2D)	(None, 29, 29, 128)	147,584
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 128)	0
dropout_1 (Dropout)	(None, 14, 14, 128)	0
conv2d_2 (Conv2D)	(None, 12, 12, 128)	147,584
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 128)	0
dropout_2 (Dropout)	(None, 6, 6, 128)	0
flatten (Flatten)	(None, 4608)	0
dense (Dense)	(None, 300)	1,382,700
batch_normalization (BatchNormalization)	(None, 300)	1,200
dropout_3 (Dropout)	(None, 300)	0
dense_1 (Dense)	(None, 2)	602

Total params: 5,048,564 (19.26 MB)
Trainable params: 1,682,654 (6.42 MB)
Non-trainable params: 600 (2.34 KB)
Optimizer params: 3,365,310 (12.84 MB)

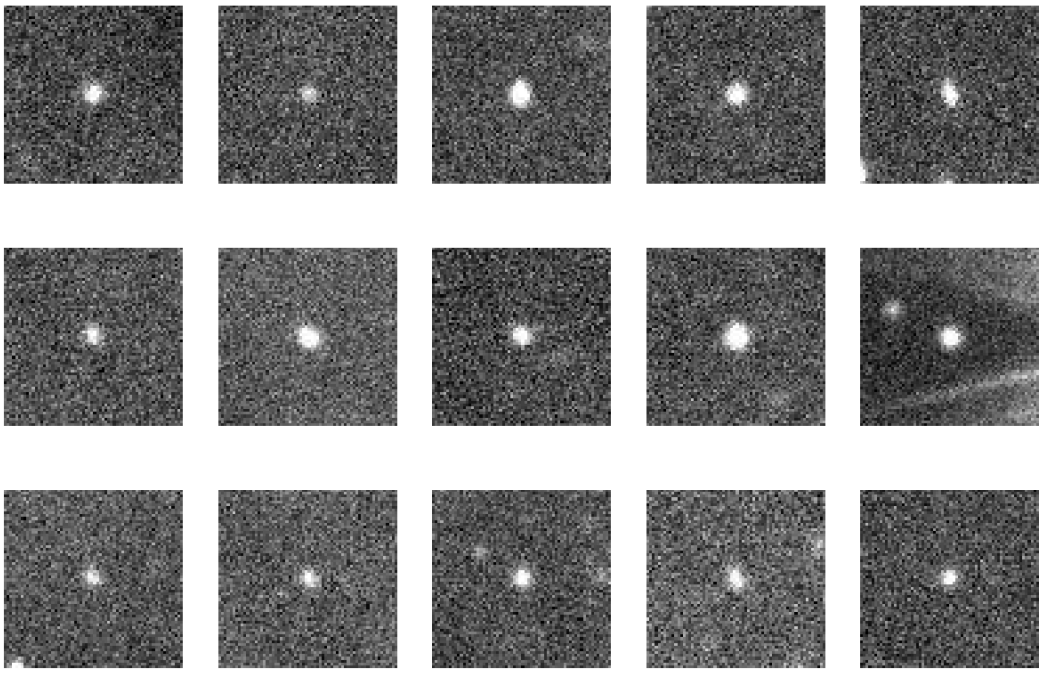




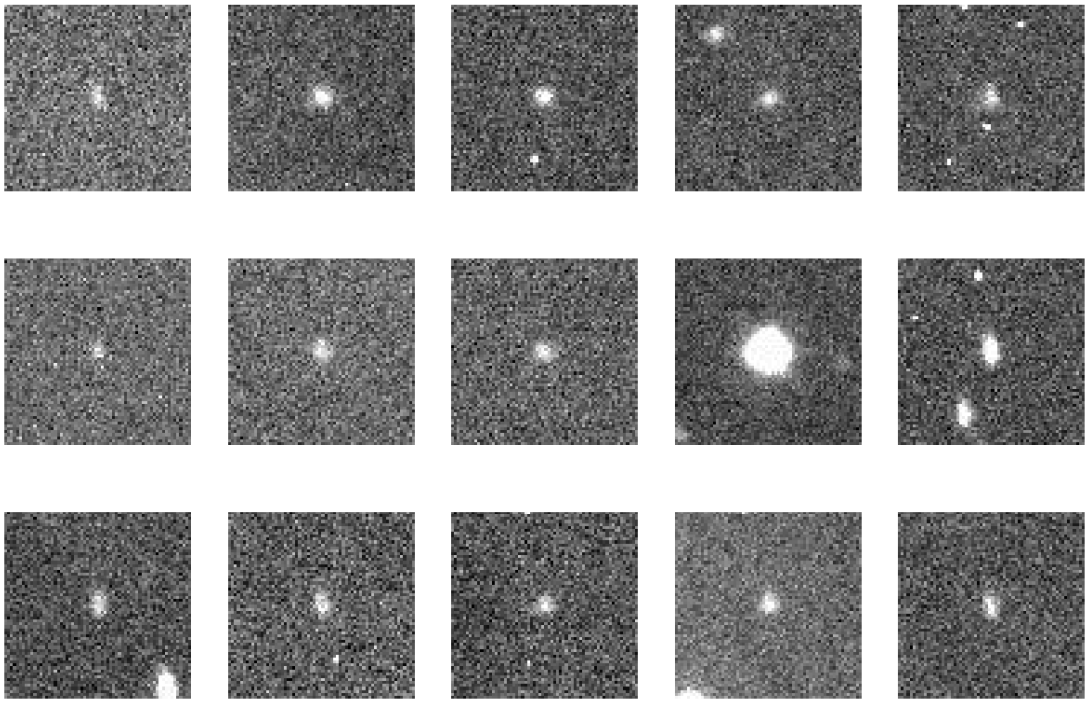
Confusion Matrix



True Label: galaxy Predicted Label: star



True Label: star Predicted Label: galaxy



Appendix J (Summary Table)

Model Architecture	Training Accuracy %	Validation Accuracy %	Testing Accuracy %
DNN with 2 hidden layers (no regularization)	52.27	48.34	48.24
DNN with 3 hidden layers (no regularization)	55.21	46.90	48.01
CNN with 2 convolution/max pooling layers (no regularization)	71.71	60.48	59.43
CNN with 3 convolution/max pooling layers (no regularization)	95.60	74.72	73.32
DNN with 2 hidden layers and L2 regularization	45.61	46.28	46.55
DNN with 3 hidden layers and L2 regularization	46.87	42.92	44.83
CNN with 2 convolution/max pooling layers and L2 regularization	76.89	63.68	63.66
CNN with 3 convolution/max pooling layers and L2 regularization	87.17	75.68	72.41
CNN with 3 convolution/max pooling layers and dropout layers of 30%	80.31	79.20	77.20

CNN with 3 convolution/max pooling layers with dropout layers of 30% and L2 regularizer (Best Model)	79.53	79.46	77.87

References

Ali, S., Shaukat, Z., Azeem, M., Sakhawat, Z., Mahmood, T., & Rehman, K. u. (2019, August). An efficient and improved scheme for handwritten digit recognition based on convolutional neural network. *Springer Link*, *1*(1125). https://link.springer.com/article/10.1007/s42452-019-1161-5

Google. (2022, July 18). *Regularization for Simplicity: L_2 Regularization | Machine Learning*. Google for Developers. Retrieved July 21, 2024, from https://developers.google.com/machine-learning/crash-course/regularization-for-simplicity/l2-regularization

Kim, H.-R. (2018, November). Building Emotional Machines: Recognizing Image Emotions Through Deep Neural Networks. *IEEE Xplore*, *20*(11), 2980-2992. https://ieeexplore.ieee.org/abstract/document/8344491